Programming Languages: Application and Interpretation

Course philosophy and motivation

- Focus on principles over syntax
- How to think about programming languages

Why Study Programming Languages?

Languages as precise thought tools

- Ultimate human–computer interface
- Understanding limits of expression

Industrial and Scientific Importance

Languages create ecosystems

- Strategic value to companies
- Enable reasoning about correctness and safety

Who Is This Course For?

Designed for the “other 90%”

- Focus on transferable principles
- Useful for students and practitioners

Beyond Paradigms

Paradigms are limiting labels

- Modern languages are multi-paradigm
- Paradigm boxes hide real behavior

Languages as Features

Languages are collections of features

- Features interact in subtle ways
- Understanding composition matters

The Standard Model of Languages (SMoL)

Common computational core

- Applies across many real languages
- Independent of paradigms

Why SMoL Matters

Foundation for performance reasoning

- Essential for concurrency and state
- Hidden assumptions in everyday programming

The Standard Implementation Plan (SImPl)

SimPl is a **systematic recipe for implementing programming languages** that treats:

It is not a tool, compiler, or framework — it is a **mental model**.

Learning Philosophy

Active learning

- Building and experimenting
- Reflection over memorization

Learning Threads

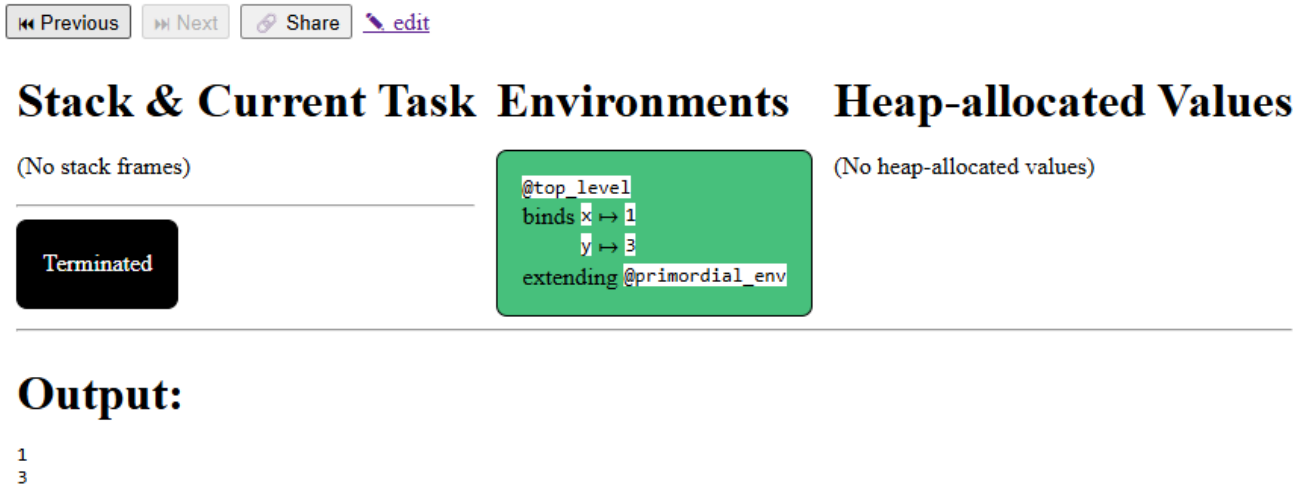
SMoL tutors

- Mystery languages
- Stacker
- Implementation
- Analysis

Role of Racket

Language-design laboratory

- Multiple languages used
- Vehicle, not the goal



Programming Languages

LOGISTIC

Teaching approach two books

Sebesta provides the formal structure

- PLAI provides the modern semantic core
- We cover Sebesta for formality, PLAI for understanding

Why Start with Sebesta?

Standard academic reference

- Shared vocabulary and categories
- Required for formality

Why Sebesta Is Not Enough

Paradigm-based organization

- Limited semantic explanation
- Modern languages blur categories

Sebesta Topics We Will Present

Programming domains

- Evaluation criteria
- Implementation methods
- Language categories

How PLAI Re-evaluates Them

Languages as features

- Semantics-first view
- Execution models matter

From Paradigms to Features

Imperative, functional, logic as history

- Modern languages are hybrid
- Features compose across paradigms

The Standard Model of Languages (SMoL)

Common semantic core

- Lexical scope, eager evaluation
- First-class functions and mutation

Evaluation: Sebesta vs PLAI

Sebesta: readability, writability, reliability

- PLAI: semantic clarity
- Trade-offs explained by semantics

Implementation Focus

Sebesta shows pipelines

- PLAI builds interpreters
- Programs as data

- Source Program
  it
- ↓
- Lexical Analysis
- ↓
- Syntax Analysis
- ↓
- Semantic Analysis
- ↓
- Intermediate Representation
- ↓
- Code Generation
- ↓
- Machine Code

Define the meaning of a language by writing an interpreter for

Powers compilers, interpreters, and LLM-based code reasoning

Why Racket

Language design laboratory

- Precise semantics
- Safe experimentat

Racket, the Programming Language

Mature | Practical | Extensible | Robust | Polished

```
#lang racket/gui

(define my-language 'English)

(define translations
  #hash([Chinese . "你好 世界"]
        [English . "Hello world"]
        [French . "Bonjour le monde"]
        [German . "Hallo Welt"]
        [Greek . "Γειά σου, κόσμε"]
        [Portuguese . "Olá mundo"]
        [Spanish . "Hola mundo"]
        [Thai . "สวัสดีชาวโลก"]
        [Turkish . "Merhaba Dünya"]))

(define my-hello-world
  (hash-ref translations my-language
            "hello world"))

(message-box "" my-hello-world)
```

Racket, the Language-Oriented Programming Language

Little Macros | General Purpose | Big Macros | Easy DSLs | IDE Support | Any Syntax

What This Means for You

Sebesta chapters are referenced

- PLAI drives reasoning
- You will build languages

Finally

Classical structure

- Modern understanding
- Deeper mastery of languages